Amendments to the Claims

The listing of claims will replace all prior versions, and listings of claims in the application.

1-19. (*canceled*)

20. (*previously presented*) A computer based method of dynamically configuring the execution of an application for a particular hardware configuration comprising:

loading an application graph of the application on a hardware configuration, wherein the application graph is a directed data flow graph wherein a node of the application graph identifies a processing block and links of the application graph indicate the direction of the data flow between processing blocks, wherein the application graph is independent of hardware configurations;

loading a dictionary of components on the hardware configuration, wherein a component represents a feature, wherein a component contains at least one implementation of the feature, wherein an implementation of the feature is represented by a set of processing blocks, wherein an implementation of the feature is customized for at least one hardware configuration;

inserting at least one set of processing blocks representing an implementation of a component into the application graph, wherein the implementation is customized for the hardware configuration; and

executing the application graph, wherein the application graph is dynamically modified, wherein said executing step includes:

traversing the application graph;

executing the processing blocks of the application graph, wherein executing the processing blocks specifies a necessary feature;

selecting a component from the dictionary based on the specified necessary feature;

selecting an implementation from the selected component based on the hardware configuration; and modifying the application graph dynamically at run time by inserting the set of processing blocks of the selected implementation into the application graph.

21. (*previously presented*) The computer based method of claim 20, wherein the traversing step segregates the processing blocks into threads, wherein the traversing step comprises:

mapping each processing block in the application graph to one of a plurality of phase of executions, wherein a phase of execution represents a subset of the application graph wherein all processing blocks in one phase of execution can be executed before or after the execution of another phase of execution;

mapping each phase of execution to one of a plurality of stage of executions, wherein all processing blocks in all phases of execution in a stage of execution can run on a single thread;

creating a block execution order list for each stage of execution, wherein a block execution order list indicates the order in which each processing block in a stage of execution will be executed; and

assigning each stage of execution to one of a plurality of threads, wherein all the processing blocks of a stage of execution will execute on the assigned thread according to the block execution order list for the stage of execution.

22. (*previously presented*) The computer based method of claim 21, wherein said creating step comprises:

creating the block execution order list for each stage of execution by ordering the processing blocks according to the frequency in which each processing block should execute.

23. (*previously presented*) The computer based method of claim 20, wherein each implementation of a feature includes a resource negotiation information, and wherein said selecting an implementation step includes:

identifying a set of implementations in the selected component that is compatible with the hardware configuration;

identifying the resources available on the hardware configuration; and

selecting the implementation from the set of implementations based on the resource negotiation information for each implementation in the set of implementations and the resources available on the hardware configuration.

24. (*new*) A system for dynamically configuring the execution of an application for a particular hardware configuration comprising:

a processor; and

a memory in communication with the processor, wherein the memory stores a plurality of processing instructions for directing the processor to:

load an application graph of the application on a hardware configuration, wherein the application graph is a directed data flow graph wherein a node of the application graph identifies a processing block and links of the application graph indicate the direction of the data flow between processing blocks, wherein the application graph is independent of hardware configurations;

load a dictionary of components on the hardware configuration, wherein a component represents a feature, wherein a component contains at least one implementation of the feature, wherein an implementation of the feature is represented by a set of processing blocks, wherein an implementation of the feature is customized for at least one hardware configuration;

insert at least one set of processing blocks representing an implementation of a component into the application graph, wherein the implementation is customized for the hardware configuration; and

execute the application graph, wherein the application graph is dynamically modified, wherein the processing instructions for directing the processor to execute the application graph include instructions for directing the processor to:

traverse the application graph;

execute the processing blocks of the application graph, wherein executing the processing blocks specifies a necessary feature;

select a component from the dictionary based on the specified necessary feature;

select an implementation from the selected component based on the hardware configuration; and

modify the application graph dynamically at run time by inserting the set of processing blocks of the selected implementation into the application graph.

25. (*new*) The system of claim 24, wherein the processing instructions for directing the processor to traverse the application graph segregates the processing blocks into threads and include instructions for directing the processor to:

map each processing block in the application graph to one of a plurality of phase of executions, wherein a phase of execution represents a subset of the application graph wherein all processing blocks in one phase of execution can be executed before or after the execution of another phase of execution;

map each phase of execution to one of a plurality of stage of executions, wherein all processing blocks in all phases of execution in a stage of execution can run on a single thread;

create a block execution order list for each stage of execution, wherein a block execution order list indicates the order in which each processing block in a stage of execution will be executed; and

assign each stage of execution to one of a plurality of threads, wherein all the processing blocks of a stage of execution will execute on the assigned thread according to the block execution order list for the stage of execution.

26. (*new*) The system of claim 25, wherein the processing instructions for directing the processor to create a block execution order list for each stage of execution include instructions for directing the processor to:

create the block execution order list for each stage of execution by ordering the processing blocks according to the frequency in which each processing block should execute.

27. (*new*) The system of claim 24, wherein each implementation of a feature includes a resource negotiation information, and wherein the processing instructions for directing the processor to select an implementation include instructions for directing the processor to:

identify a set of implementations in the selected component that is compatible with the hardware configuration;

identify the resources available on the hardware configuration; and

select the implementation from the set of implementations based on the resource negotiation information for each implementation in the set of implementations and the resources available on the hardware configuration.

28. (*new*) A computer program product comprising a tangible computer usable medium having control logic stored therein for causing a computer to dynamically configure the execution of an application for a particular hardware configuration, said control logic comprising:

first computer readable program code means for causing the computer to load an application graph of the application on a hardware configuration, wherein the application graph is a directed data flow graph wherein a node of the application graph identifies a processing block and links of the application graph indicate the direction of the data flow between processing blocks, wherein the application graph is independent of hardware configurations;

second computer readable program code means for causing the computer to load a dictionary of components on the hardware configuration, wherein a component represents a feature, wherein a component contains at least one implementation of the feature, wherein an implementation of the feature is represented by a set of processing blocks, wherein an implementation of the feature is customized for at least one hardware configuration;

third computer readable program code means for causing the computer to insert at least one set of processing blocks representing an implementation of a component into the application graph, wherein the implementation is customized for the hardware configuration; and

fourth computer readable program code means for causing the computer to execute the application graph, wherein the application graph is dynamically modified, wherein said fourth computer readable program code means includes:

fifth computer readable program code means for causing the computer to traverse the application graph;

sixth computer readable program code means for causing the computer to execute the processing blocks of the application graph, wherein executing the processing blocks specifies a necessary feature;

seventh computer readable program code means for causing the computer to select a component from the dictionary based on the specified necessary feature;

eighth computer readable program code means for causing the computer to select an implementation from the selected component based on the hardware configuration; and

ninth computer readable program code means for causing the computer to modify the application graph dynamically at run time by inserting the set of processing blocks of the selected implementation into the application graph.

29. (*new*) The computer program product of claim 28, wherein the fifth computer readable program code means segregates the processing blocks into threads and comprises:

eleventh computer readable program code means for causing the computer to map each processing block in the application graph to one of a plurality of phase of executions, wherein a phase of execution represents a subset of the application graph wherein all processing blocks in one phase of execution can be executed before or after the execution of another phase of execution;

twelfth computer readable program code means for causing the computer to map each phase of execution to one of a plurality of stage of executions, wherein all processing blocks in all phases of execution in a stage of execution can run on a single thread;

thirteenth computer readable program code means for causing the computer to create a block execution order list for each stage of execution, wherein a block execution order list indicates the order in which each processing block in a stage of execution will be executed; and

fourteenth computer readable program code means for causing the computer to assign each stage of execution to one of a plurality of threads, wherein all the processing blocks of a stage of execution will execute on the assigned thread according to the block execution order list for the stage of execution.

30. (*new*) The computer program product of claim 29, wherein the thirteenth computer readable program code means comprises:

fifteenth computer readable program code means for causing the computer to create the block execution order list for each stage of execution by ordering the processing blocks according to the frequency in which each processing block should execute.

31. (*new*) The computer program product of claim 28, wherein each implementation of a feature includes a resource negotiation information, and wherein the eighth computer readable program code means comprises:

eleventh computer readable program code means for causing the computer to identify a set of implementations in the selected component that is compatible with the hardware configuration;

twelfth eleventh computer readable program code means for causing the computer to identify the resources available on the hardware configuration; and thirteenth computer readable program code means for causing the computer to select the implementation from the set of implementations based on the resource negotiation information for each implementation in the set of implementations and the resources available on the hardware configuration.